User's Guide
Version 1.0 - October 2013

AMBA DMA Controller

With 32-bit/64-bit AXI Support

User's Guide Generic / ASIC

User's Guide
Version 1.0 - October 2013

Contents

User's Guide
Version 1.0 - October 2013

Document Founder

Mr. Paul. P. Bates	*Nine Ways Research & Development Ltd (UK)*
Written under the governance and agreement of **Mr. Daniel Koehler**	*MorethanIP GmbH*

About This Release Document

This document describes specifically coding relevancies, package options and internal architecture from the HDL perspective.

Intended Audience

This document is written for associates and employees of MorethanIP GmbH, for customers/clients using the hardware and integrating into project designs. It is a full open release document that is available with the purchase and deliverable of the AMBA DMA Controller product.

List of Tables

List of Figures

1 Design Kit Introduction

The DMA Controller package is agnostic and generic. That is – it can be synthesized and instantiated into different design IDE programs for different device families even though it as a deliverable comes packaged in a defined and simple structure.

This structure is described and listed in detail in the following section of this document.

Despite the many different variations of configuration (all changeable in the *package.verilog file*) there is only one set of source files and simulation scripts. Usually an IP core package may include different duplicate files some for *32-bit* and some *64-bit* deviations. Moreover, there can often be a JAVA application to allow GUI configuration that then produces specific package source folders with hardcoded parameters and constants. However, this package does include such a configurator IDE.

This package for the AMBA DMA Controller is monolithic – that being, all configuration options for both synthesis and ModelSIM Testbench simulation are kept in the <*package.verilog*> file included by all source HDL (located in *source/package/verilog*). This allows for a single ModelSIM script and also only a single source directory with generic source files capable of being synthesized in a whole variation of different configurations and sizes.

It will be described, *later in this document*, how the simulation using ModelSIM is performed by setting the parameters of operation in the *package.verilog* file, then executing a simple script for ModelSIM.

Also, it will show how the instantiation of this *package* in a synthesis environment such as Libero, is simply the IDE copying the source directory files into its own instance directory in the IDE. At this point, changes to the *package.verilog* file in the newly copied and separate DMA instance directory in the synthesizer IDE will only affect that instance. It will not affect this initial package directory content.

Furthermore, if multiple instantiations of the same AMBA DMA Controller occur, then the package.verilog file in separate directories is the only duplication needed per instance which then leads to the IDE referring to the different <*package.verilog*> header file.

2 *Design Flow*

2.1 Overview

The different steps of the AMBA DMA Controller Design and hardware core's usage within the provided test environment are shown in the following paragraphs and include:

- *Core generation*
- *RTL Simulation*
- *Synthesis and Implementation*

The full design kit provides scripts for ease of use and fast design, verification and implementation turn-around.

2.2 Recommended Tools

Table 1: Recommended Tools

Design Flow	***Tool***	***Version***
Simulation	ModelSIM SE	6.x or Higher
	ModelSIM PE	
	ModelSIM AE	
Synthesis (FPGA)	Synopsis Synplify Pro	ME G-2012.09MSP1 or higher
Synthesis	Synopsys Design Compiler	10.9.1 or Higher

2 DMA Controller Core Database

After the design is generated, the root directory of the DMA Controller package is called **AMBA_DMA_Controller** which contains a documentation directory (containing these documents), **libero, quartus, simulation, source, synthesis** which houses all of the IP core hardware source, simulation and test-bench facilities, and finally the **LinuxDriver** directory.

Table 2: Design Kit Directories Description

Directory	Description
libero	Contains the <*MakeLibero*> file script that copies the agnostic generic driver in this package into a SmartFusion MicroSemi project HDL directory. **Note**: You *will need to edit this file to select the target directory*
quartus	Altera Quartus project example that synthesizes the core (if available – and may be empty as a directory)
simulation	Scripts to configure and execute simulation.
source	Design and Testbench Source Files. The Testbench sub-directory contains the DDR-AXI Slave ARM model for verification of the ARM sub-system memory controller.
synthesis	*Synopsys Synthesis Scripts.* An example implementation script is provided to show the list of files and their order. If you have the Synopsis/Synplify package installed on your computer, executing the <*SynopsisLint.bat*> file will perform a full scale synthesis check on the DMA package as stand-alone. **Note:** *You must have Synopsis tools installed on your PC running Windows XP or higher.* *Linting.* <LintCheck.bat> file in this directory will perform a standard generic linting analysis using Verilator. **Note:** *You must have Verilator installed on your PC running Windows XP or higher.*
documentation	For the customer package, this directory is at the same level as the above and contains all of the PDF documents for this product.
LinuxDriver	Contains the entire sub-tree of all the C code, build files, make script and binaries of the device driver software.

3 Simulation Environment

A system Testbench (see Figure 1 - Testbench Setup Overview) is provided which implements the AMBA DMA Controller core with simulation control state machines, MAC FIFO and DDR-AXI model drivers / monitors which generate and analyze traffic to and from the DMA Controller core.

Table 3: Testbench Files

File	**Directory**	**Description**
testbench.v	source/testbench/verilog	Testbench using the DMA top-level
testbench_functions.v	source/testbench/verilog	TB lower level functions
tb_model_host.v	source/testbench/verilog	External <reg> memory block
tb_ddraxi_slave	source/testbench/verilog	MDDR AXI Slave model (ARM)
tb_signals.v	source/testbench/verilog	All reg, wire and integer definitions
tb_clocks	source/testbench/verilog	Clock generation block
testmac64 ***<directory>***	source/testbench/verilog	Instantiated MAC for loop-back testing

- **External Register Model Host**: This emulates another IP core cascaded onto the external register bus of the DMA Controller. The external register bus allows for other IP cores to cascade off the primary DMA Controller and be mapped into the same AHB-Lite addressing scheme from a processor sub-system. The Testbench performs a series of contiguous read/write transactions to this model to check that the DMA HDL is operating well.
- **MDDR AXI Slave:** This model emulates an MDDR or FDDR AXI Slave interface that the DMA Controller will be expected to connect with. The AXI Master in the DMA Controller in the Testbench is controlled to read and write Ethernet frames to/from the AXI model.

The Testbench has three possible configurations:

1. *32-bit AMBA data width*.
2. *64-bit AMBA data width*.
3. *Variable descriptor table sizes for RX and TX*: The four descriptor tables can have their sizes adjusted in the <*package.verilog*> file along with the *32-bit* and *64-bit* modes (*above*).

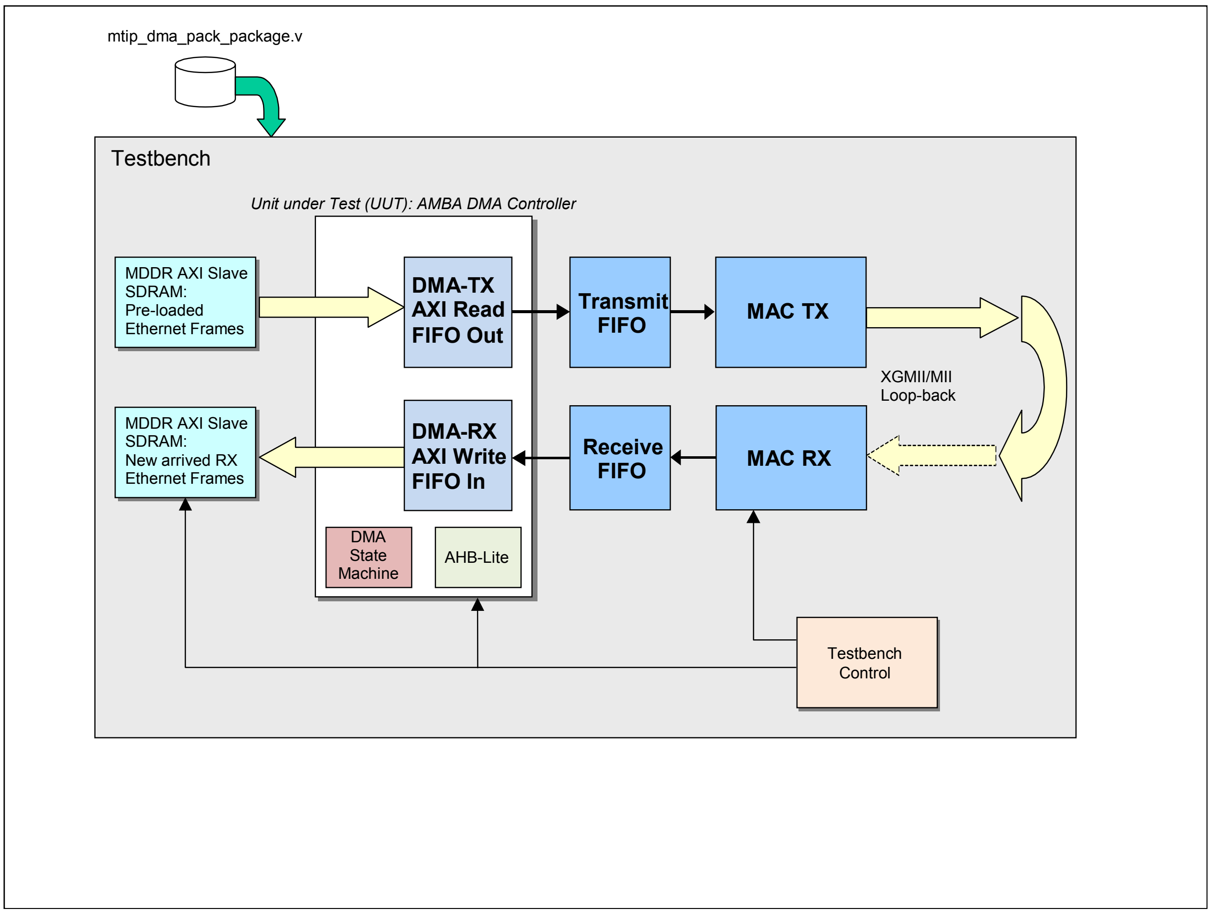


Figure 1 - Testbench Setup Overview

The Testbench includes several configuration and scenario files to exercise the core.

3.1 Running Simulation Using ModelSIM

3.1.1 Overview

The AMBA DMA Controller core files and dependencies are available in the file <*comp_amba_dma.do*> (*simulation* directory), the Testbench file and simulation models files are listed in the files <*sim_amba_dma.do*>.

To perform simulation, the following steps should be followed:

1. Change to the *simulation* directory.
2. Compile the core database: <*comp_amba_dma.do*>
3. Compile the simulation database and run simulation: <*sim_amba_dma.do*>

Note that the Testbench will produce many debugging information during simulation. There are errors and warning messages during the simulation, which are intended (e.g. purposely generated errors). When the Testbench stops the output must show *"* Simulation Ends without Error *"* at the end indicating a successful pass through all scenarios.

3.1.2 Simulation Options

The simulation is controlled by a set of `**define** parameters. These parameters are given to the compiler when compiling the Testbench. See the script file <*sim_amba_dma.do*> for a list of available options (see +define+ statements).

3.1.3 Package Options

The simulation is also heavily influenced and controlled by the main *package.verilog* header file for the main source HDL under test. The <*testbench.v*> also observes and uses these parameters and `defines.

For instance, if you comment out or uncomment the **`define MTIP_MAC_UUT_64BIT** then it will change the simulation scenario between *32-bit* and *64-bit* data widths for the AMBA system (FIFO and AXI)

Also, you change the parameters for the TX and TX Descriptor sizes;

```
parameter   RXCH0_DESCSIZE    = 4;    // Customer choice: Size of RX channel 0 Descriptors
parameter   RXCH1_DESCSIZE    = 4;    // Customer choice: Size of RX channel 1 Descriptors
parameter   TXCH0_DESCSIZE    = 4;    // Customer choice: Size of TX channel 0 Descriptors
parameter   TXCH1_DESCSIZE    = 4;    // Customer choice: Size of TX channel 1 Descriptors
```

This will change the scenario for the Testbench as well. The DMA controller is monolithic and so changes to the main *package.verilog* do not change just the package HDL source under test but also the Testbench simulation. This all adds to the simplicity of the whole operation.

Note: *The remaining parameters in the package.verilog file can also be changed to help test the widest possible set of scenarios possible.*

4 *Core Usage*

4.1 Core Configuration and Synthesis options

4.1.1 Global Synthesis Options

The following table lists the relevant synthesis options found in the so-called global package file. This file is included by all sources to set global definitions. These are for size optimizations and to include/exclude features that affect pin-outs of modules.

If multiple different Cores are synthesized together, all their "*common_header.verilog*" files need to be merged into a single file of that name containing all settings of all cores.

The package file is located at:

- **source/package/verilog/common_header.verilog**.

The following settings are available for change. All others must not be changed.

Table 4: Global Package Definitions (common_header)

Setting Name	Description
No Relevant Values	N/A

4.1.2 Core Configuration Package File

The core is configured using the package file:

- **source/package/verilog/mtip_dma_pack_package.verilog**

The file is included by every relevant source file to configure several options during synthesis. The following settings in this file can be changed. All others must remain unchanged.

Table 5: Synthesis Package Definitions (mtip_dma_pack_package.verilog)

Setting Name	Type	Description	Default
		Revision Configuration	
MTIP_CUSTOMER_ SPECIFIC_REVISION	**parameter**	This can be any *16-bit* integer value. It is stated in hexadecimal in the package file and is used as a sub-version number by a project integrator using this DMA Controller package in their design.	0001 *(HEX)*
MTIP_CORE_VERSION	**parameter**	**DO NOT CHANGE**	1301 *(HEX)*
		Data Width & AXI Master Configuration	
AMBA_ADDR_WIDTH	**parameter**	Global AMBA system data width. This affects the AXI address width only in this case. It sets the number of bit of the AXI Master	32

		address bus and **MUST** match the width of the system memory bus of the host device system architecture. **Note:** *Must be a power of 2, at least 32.*	
MTIP_MAC_UUT_64BIT	**`define**	This define states whether the DMA system data width is *64-bit* or not. By `defining it declares the parameter *AMBA_DATA_WIDTH* to be 64. Else, it allows parameter *AMBA_DATA_WIDTH* to be 32. This parameter governs the width of the FIFO data interface and the AXI Master data width for both RX and TX pathways.  The parameter *AMBA_DATA_WIDTH* is controlled by a `define because of a ModelSIM Testbench requirement (see later section on Testbench) **Note:** *The width of the DMA system data width (AXI and FIFO) is restricted to 32 or 64 only. Other attempted values will cause problems with operation.*	Defined
AXI_ID_WIDTH	**parameter**	Sets the number of bits for the write AXI address ID, write AXI data ID, read AXI address ID and read AXI data ID (rid). This width is variable and you can use this to match the size of the AXI Slave that the DMA Controller Master connects to. **Note:** *Usual values are 2, 3 or 4.*	4
AXI_RXDMA_ BACKPRESSURE_SIZE	**parameter**	Depth of the RX-DMA backpressure FIFO. This FIFO cushions the initial [*sop*] data to activate the RX mechanism, and also allows for the MAC to react slowly to an RX-DMA ready signal de-assertion. This, if set too low (less than 4 can cause problems with the RX if the ready signal is dropped to the MAC). **Note:** *This can be set as high as 2048 but optimally is operates well at 16 or 32.*	32
RX & TX Descriptor Table Configuration			
RXCH0_DESCSIZE	**parameter**	Sets the size of the High priority RX-DMA (Channel 0) descriptor table size in terms of entries. This value if **<zero>** effectively disables the RX-DMA CH0 mechanism. Maximum allowed value is 256. **Note**: *Values in the higher range provide for better burst and high throughput performance, but may not synthesize well in smaller device families.*	4
RXCH1_DESCSIZE	**parameter**	Sets the size of the Low priority RX-DMA (Channel 1) descriptor table size in terms of entries.	4

		This value if <**zero**> effectively disables the RX-DMA CH1 mechanism. Maximum allowed value is 256. **Note**: *Values in the higher range provide for better burst and high throughput performance, but may not synthesize well in smaller device families.*	
TXCH0_DESCSIZE	**parameter**	Sets the size of the High priority TX-DMA (Channel 0) descriptor table size in terms of entries. This value if <**zero**> effectively disables the TX-DMA CH0 mechanism. Maximum allowed value is 256. **Note**: *Values in the higher range provide for better burst and high throughput performance, but may not synthesize well in smaller device families.*	4
TXCH1_DESCSIZE	**parameter**	Sets the size of the Low priority TX-DMA (Channel 1) descriptor table size in terms of entries. This value if <**zero**> effectively disables the TX-DMA CH1 mechanism. Maximum allowed value is 256. **Note**: *Values in the higher range provide for better burst and high throughput performance, but may not synthesize well in smaller device families.*	4
Other Settings		*All other settings **must** be left unchanged.*	N/A

4.2 Top-Level

4.2.1 Files

Table 6: Top-Level Files

File	Location	Description
mtip_amba_dma_controller.v	source/top/verilog/	The DMA Controller top-level universal for all configurations and variations.

Note that all files rely on packages included from the directory `source/package/verilog`. Hence this directory must be added to the include path of the simulator and synthesizer tools.

4.2.2 Instantiation Parameters

The top-level does *NOT* have any options that can be configured during instantiation. All parameters are controlled in the *package.verilog* file.

Instantiation would typically jus take the following form (with no parameters but just wire definitions):

```
module mtip_amba_dma_controller(
        .reset_hclk( …… ),
        .hclk( …… ),
        .reset_aclk( …… ),
        .aclk( …… ),
        .reg_rd( …… ),
        .reg_wr( …… ),
        .reg_addr( …… ),
        .reg_data_in( …… ),
        .reg_data_out( …… ),
        .reg_busy( …… ),
        .extint_inputs( …… ),
        .extint_acks( …… ),
        .hsel( …… ),
        .haddr( …… ),
        .hwrite( …… ),
        .htrans( …… ),
        .hsize( …… ),
        .hburst( …… ),
        .hwdata( …… ),
        .hprot( …… ),
        .hready( …… ),
        .hmastlock( …… ),
        .hreadyout( …… ),
        .hresp( …… ),
        .hrdata( …… ),
        .awlen( …… ),
        .awsize( …… ),
        .awburst( …… ),
        .awlock( …… ),
        .awvalid( …… ),
        .awready( …… ),
        .wvalid( …… ),
        .wlast( …… ),
        .wready( …… ),
        .bresp( …… ),
        .bvalid( …… ),
        .bready( …… ),
        .arvalid( …… ),
        .arlen( …… ),
        .arsize( …… ),
        .arburst( …… ),
        .arlock( …… ),
        .arready( …… ),
        .rlast( …… ),
        .rresp( …… ),
         .rvalid( …… ),
        .rready( …… ),
        .araddr( …… ),
        .awaddr( …… ),
        .wdata( …… ),
        .wstrb( …… ),
        .awid( …… ),
        .wid( …… ),
        .bid( …… ),
        .arid( …… ),
        .rid( …… ),
        .rdata( …… ),
        .rmw( …… ),
        .ff_tx_dsav( …… ),
        .ff_tx_data( …… ),
        .ff_tx_sop( …… ),
        .ff_tx_eop( …… ),
        .ff_tx_err( …… ),
        .ff_tx_dval( …… ),
        .ff_tx_mod( …… ),
        .ff_tx_xstat( …… ),
        .ff_tx_crc_fwd( …… ),
        .ff_tx_protocol_checksum_enable( …… ),
        .ff_tx_protocol_checksum( …… ),
        .ff_tx_rdy( …… ),
        .ff_tx_ch( …… ),
        .ff_tx_septy( …… ),
        .ff_rx_protocol_checksum_valid( …… ),
        .ff_rx_err_stat( …… ),
        .ff_rx_data( …… ),
        .ff_rx_sop( …… ),
        .ff_rx_eop( …… ),
        .ff_rx_err( …… ),
        .ff_rx_mod( …… ),
        .ff_rx_xstat( …… ),
        .ff_rx_dval( …… ),
        .ff_rx_rdy( …… ),
        .ff_rx_dsav( …… ),
        .ff_rx_ch( …… ),
        .ff_rx_protocol_checksum( …… ),
        .cpu_int( …… ),
        .ext_resetn( …… )
);
```

4.3 Simulation

To simulate the AMBA DMA Controller when implemented in a custom project, compile the HDL source core in the design work directory as shown in the script <*simulation/comp_amba_dma.do*>

Behavioral models (external register device, MDDR AXI Slave, AnySpeed MAC32/64) are provided in verilog source in the <*source/testbench/verilog*> directory. The models can be re-used in a custom project Testbench and provide a robust and flexible verification tool.

Table 7: Behavioral Models Summary

Verilog Model	***Location***	***Description***
tb_model_host.v	source/testbench/verilog	Simulates another MorethanIP hardware core cascaded off the DMA Controller using the external register bus for the IO Register of that 3rd party core.
tb_ddraxi_slave.v	source/testbench/verilog	Behaves according to the standard AMBA ARM memory bus sub-system MDDR that exposes an AXI Slave interface that connects to the DMS Controller AXI Master.
<testmac64> *(directory containing verilog HDL source)*	*source/testbench/verilog/testmac64/*	MorethanIP independent well-established 32/64 MAC. This allows for the frame on the TX of the DMA Controller to be looped-back through a MAC core into the RX of the DMA Controller core. This is the primary function of the Testbench simulation.

4.4 Memory Implementation

4.4.1 Internal memory structures

The AMBA DMA Controller has only one instance of an implementation of memory structures. This is the back pressure FIFO in the RX-DMA (AXI write) inner-block. This is used to allow the RX of the MAC FIFO to burst some transaction words whilst the AXI write bus is setting up. Moreover, if there are many scenarios whereby the AXI memory system experiences in a large complex system slowdowns and delays due to bus contention – then the FIFO allows the MAC to continue transactions for a short while by filling the FIFO whilst the AXI bus waits.

It is widely seen that the MAC cores do take a finite number of clock cycles to respond to *ff_rx_rdy* being de-asserted by the DMA Controller – which gives rise for the need for the RX-DMA FIFO.

Note: *The RX backpressure FIFO is ideally around 32 words in depth.*

This is enough to handle:

1. The MAC slow response *to ff_rx_rdy.*
2. The time between an *SOP* seen and the AXI bus activated for write transactions
3. The start of a next Ethernet frame from a MAC whilst the previous AXI bus transaction completes
4. Not to use too much hardware look-up tables in the Fabric/ASIC.

The files describing memories used in the design are:

Table 8: Memory Descriptions

Directory/File	**Description**
source/aximaster/verilog/ mtip_aximaster_fifo.v	Single-Clock Dual-Port Memory. Used for the RX-DMA backpressure FIFO in *mtip_aximaster_wr.v* To change the depth, use the parameter : AXI_RXDMA_BACKPRESSURE_SIZE in *package.verilog*

5 Synthesis

5.1 Synopsis Synthesis

The <*synthesis*> directory contains a synthesis script <*SynopsisLint.prj*> example used to synthesize the design. Synthesis scripts are provided for *Synopsys Synplify Pro* as a starting point. The script needs to be updated to the proper library and other environmental options as necessary.

For synthesis it is important to have the directory `source/package/verilog` added to the include search path of the compiler as most of the files include the package files found in that directory during synthesis.

Note: *You must have the Synopsis Synplify IDE and tools installed on your PC. This is not included with the deliverable DMA Controller package as this is far too large file size.*

5.2 Verilator Lint Synthesis

The <*synthesis*> directory also contains a Linting script <*Lintcheck.bat*> for Windows. Use this to check that the generic syntax and verilog coding is good. You can edit the *Lintcheck.bat* file of you want to check other modules or you create different top modules for this project.

Important Note: *You must have Verilator copied into the root directory of your computer as \Verilator\ with the following files:*

cygwin1.dll
verilator
verilator.bin
verilator_compiled_win

You can obtain the Verilator package from Nine Ways Research & Development Ltd. See website www.nineways.co.uk to contact us and obtain the file.

Note: *This is not included in the deliverable package of the AMBA DMA Controller as the file sizes are too big and bloats the content.*

5.3 Libero Synthesis from Actel/Microsemi

As a start point, the directory within the package called <*libero*> contains a Windows BATCH file script <*MakeLibero.bat*> to directly install the source HDL content of the DMA Controller into a Microsemi project. This then is ready for instantiation and use within an *FPGA/ASIC* IDE environment.

This is very specific to the Microsemi environment and only acts as a start point. Over time, more scripts can be added that will help end users install the HDL source into specific IDE environments.

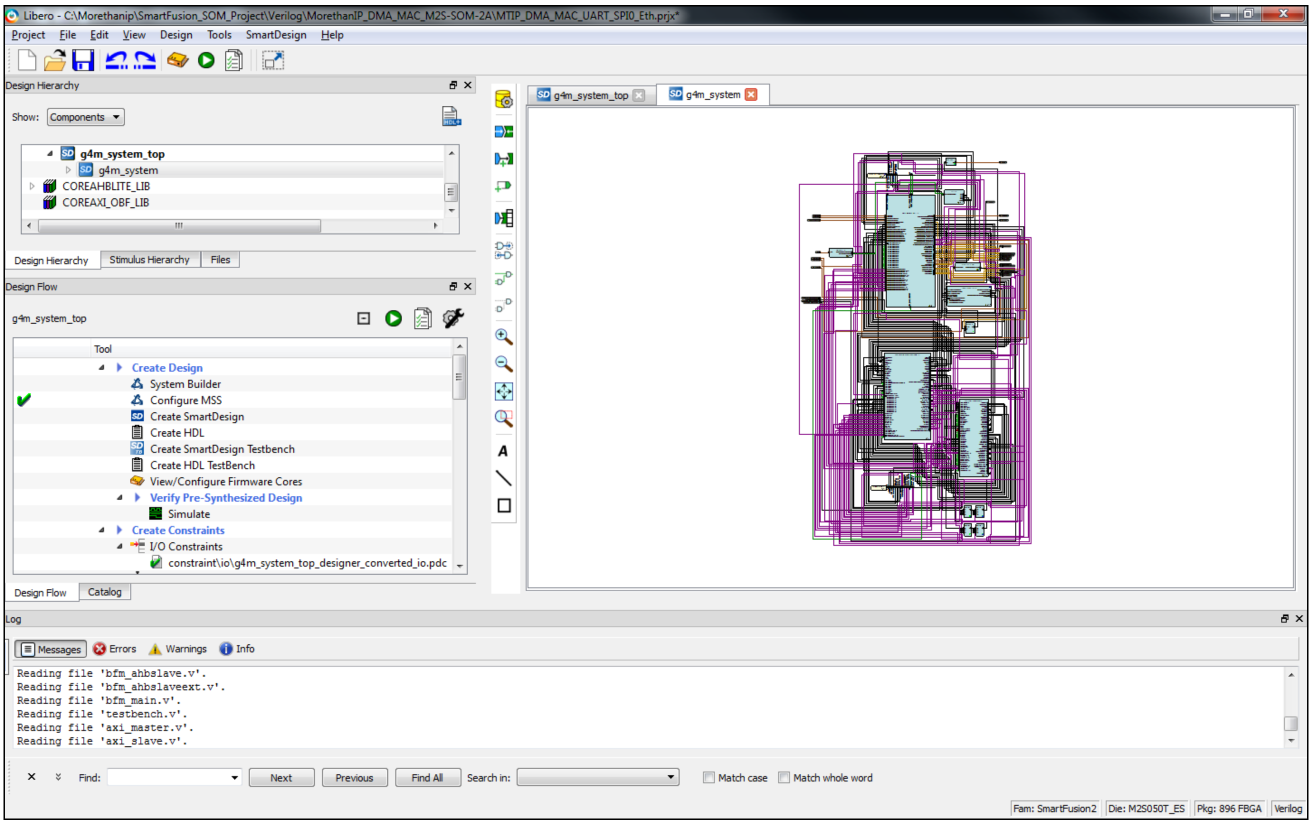
The IDE package for Microsemi is typically called **Libero Pro** or similar derivatives. However, this script is setup specifically for if you are using Nine Ways Research & Developments start-up project that provides a SmartFusion2 environment with the DMA Controller and MorethanIP's MAC as fabric cores.

This is available from Nine Ways as an SVN-repository check-out (*Subversion*). Contact details can be obtained from www.nineways.co.uk

If you wish to have your own Libero project then where ever this is located in your file system on your PC, you will have to edit the file <*MakeLibero.bat*> in the <*libero*> directory.

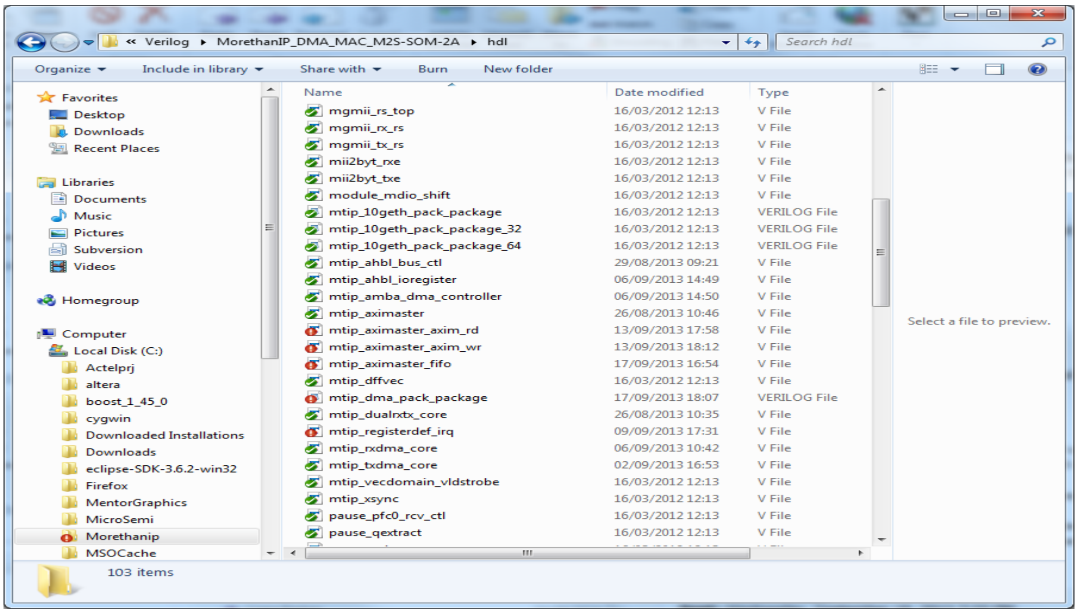
Once you have installed the files into the Libero system, they are separate entities along with the other core verilog files in that target Microsemi directory. They can then have their own **parameter** settings and **`define** configuration as discussed previously in this document.

Figure 2 - Libero from Microsemi with the Nine Ways start-up project open


Libero - C:\Morethanip\SmartFusion_SOM_Project\Verilog\MorethanIP_DMA_MAC_M2S-SOM-2A\MTIP_DMA_MAC_UART_SPI0_Eth.prjx*
Project File Edit View Design Tools SmartDesign Help
Design Hierarchy
Show: Components
g4m_system_top
g4m_system
COREAHBLITE_LIB
COREAXI_OBF_LIB
Design Hierarchy Stimulus Hierarchy Files
Design Flow
g4m_system_top
Tool
Create Design
System Builder
Configure MSS
Create SmartDesign
Create HDL
Create SmartDesign Testbench
Create HDL TestBench
View/Configure Firmware Cores
Verify Pre-Synthesized Design
Simulate
Create Constraints
I/O Constraints
constraint\io\g4m_system_top_designer_converted_io.pdc
Design Flow Catalog
g4m_system_top
g4m_system
Log
Messages Errors Warnings Info
Find: Next Previous Find All Search in: Match case Match whole word
Fam: SmartFusion2 Die: M2S050T_ES Pkg: 896 FBGA Verilog

```
Reading file 'bfm_ahbslave.v'.
Reading file 'bfm_ahbslaveext.v'.
Reading file 'bfm_main.v'.
Reading file 'testbench.v'.
Reading file 'axi_master.v'.
Reading file 'axi_slave.v'.
```

Figure 3 - Directory in the DMA project for SmartFusion2 start-up, where HDL files were copied



Name	Date modified	Type
mgmii_rs_top	16/03/2012 12:13	V File
mgmii_rx_rs	16/03/2012 12:13	V File
mgmii_tx_rs	16/03/2012 12:13	V File
mii2byt_rxe	16/03/2012 12:13	V File
mii2byt_txe	16/03/2012 12:13	V File
module_mdio_shift	16/03/2012 12:13	V File
mtip_10geth_pack_package	16/03/2012 12:13	VERILOG File
mtip_10geth_pack_package_32	16/03/2012 12:13	VERILOG File
mtip_10geth_pack_package_64	16/03/2012 12:13	VERILOG File
mtip_ahbl_bus_ctl	29/08/2013 09:21	V File
mtip_ahbl_ioregister	06/09/2013 14:49	V File
mtip_amba_dma_controller	06/09/2013 14:50	V File
mtip_aximaster	26/08/2013 10:46	V File
mtip_aximaster_axim_rd	13/09/2013 17:58	V File
mtip_aximaster_axim_wr	13/09/2013 18:12	V File
mtip_aximaster_fifo	17/09/2013 16:54	V File
mtip_dffvec	16/03/2012 12:13	V File
mtip_dma_pack_package	17/09/2013 18:07	VERILOG File
mtip_dualrxtx_core	26/08/2013 10:35	V File
mtip_registerdef_irq	09/09/2013 17:31	V File
mtip_rxdma_core	06/09/2013 10:42	V File
mtip_txdma_core	02/09/2013 16:53	V File
mtip_vecdomain_vldstrobe	16/03/2012 12:13	V File
mtip_xsync	16/03/2012 12:13	V File
pause_pfc0_rcv_ctl	16/03/2012 12:13	V File
pause_qextract	16/03/2012 12:13	V File

6 Contact

MorethanIP GmbH	E-Mail : **info@morethanip.com** Internet : **www.morethanip.com** **Europe** Muenchner Str. 199 D-85757 Karlsfeld Germany Tel : **+49 (0) 8131 333939 0** FAX : **+49 (0) 8131 333939 1**
Nine Ways Research & Development Ltd	E-Mail : **pbates@nineways.co.uk** Internet : **www.nineways.co.uk** **UK** Unit G.15, iDCentre, Lathkill House, rtc Business Park London Road, Derby. DE24 8UP United Kingdom Tel : **+44 (0) 1332 258847** FAX : **+44 (0) 1332 258823**

7 *Document History*

Document Change Notices (DCO)

Version	Description	Created/Changed By	Date
Version 1.0	Initial release as according to Version 1.0 of the Verilog coding package	Paul Bates: Nine Ways R&D (UK) Ltd	**31st October 2013**